

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-152881

(43) 公開日 平成8年(1996)6月11日

(51) Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所
G 1 0 H 1/00	1 0 2 Z			
	Z			
G 1 0 K 15/04	3 0 2 D			

審査請求 未請求 請求項の数 5 F D (全 9 頁)

(21) 出願番号 特願平6-319337

(22) 出願日 平成6年(1994)11月29日

(71) 出願人 000004329

日本ビクター株式会社

神奈川県横浜市神奈川区守屋町3丁目12番
地

(72) 発明者 矢戸 一郎

神奈川県横浜市神奈川区守屋町3丁目12番
地 日本ビクター株式会社内

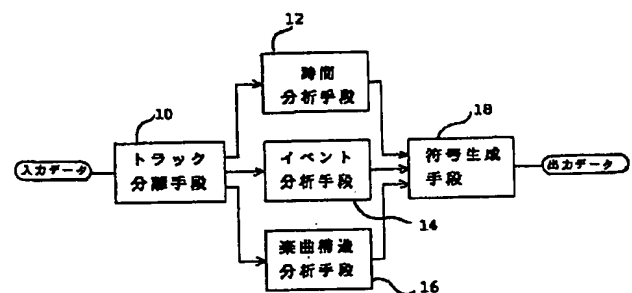
(74) 代理人 弁理士 二瓶 正敬

(54) 【発明の名称】 演奏情報圧縮装置

(57) 【要約】

【目的】 時間情報とイベント情報を有する演奏情報データの性質を利用して、データ量を大きく削減することのできる演奏情報圧縮装置を提供する。

【構成】 楽音を構成する音の発音時間情報と共に音の音程その他を指定するイベントを含む演奏情報にตอบสนองし、演奏情報のイベント間の相対時間を分析する時間分析手段12と、演奏情報にตอบสนองし、イベントの種類を分析するイベント分析手段14と、演奏情報にตอบสนองし、所定時間間隔でデータを分割してセグメント化し、各々のセグメントから同一内容のデータを検出する楽曲構造分析手段16と、時間分析手段とイベント分析手段と楽曲構造分析手段のそれぞれの分析結果に基づいて演奏情報を符号化する符号生成手段18とを、有する。



【特許請求の範囲】

【請求項1】 楽音を構成する音の発音時間情報と共に音の音程その他を指定するイベントを含む演奏情報に
1 応答し、前記演奏情報のイベント間の相対時間を分析する時間分析手段と、
前記演奏情報に
2 応答し、イベントの種類を分析するイベント分析手段と、
前記演奏情報に
3 応答し、所定時間間隔でデータを分割してセグメント化し、各々のセグメントから同一内容のデータを検出する楽曲構造分析手段と、
前記時間分析手段と前記イベント分析手段と前記楽曲構造分析手段のそれぞれの分析結果に基づいて前記演奏情報を符号化する符号生成手段とを、
有する演奏情報圧縮装置。

【請求項2】 前記時間分析手段がイベント間の相対時間の出現頻度を算出する手段と、出現するイベント間の相対時間の最大公約数を算出する手段とを有する請求項1記載の演奏情報圧縮装置。

【請求項3】 前記イベント分析手段が特定の種類のイベントの出現の頻度を算出する手段を有する請求項1記載の演奏情報圧縮装置。

【請求項4】 前記イベント分析手段が全ての種類のイベントの出現の頻度を算出する手段を有する請求項1記載の演奏情報圧縮装置。

【請求項5】 前記イベント分析手段が特定の種類のイベントが所定数以上連続している部分を検出する手段を有する請求項1記載の演奏情報圧縮装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は、演奏情報のデータ量を削減する演奏情報圧縮装置に関する。

【0002】

【従来の技術】 一般に通信カラオケと言われるシステムが登場している。これは従来の光ディスク等の記憶媒体に記録された楽音、歌詩、画像等のデータをリクエストに応じて選択して再生するものに代えて、ステーション（ホスト）と各端末を通信回線で結び、新譜を短時間で端末に送る等の配信を可能としたものである。かかる通信カラオケでは、MIDI信号のデータが用いられ、楽音信号は楽音を構成する音の音程と発音時間等を指定する演奏情報として送信されている。従来、MIDIデータの保存形式として、スタンダード MIDI ファイル（以下SMFとする）と呼ばれる形式が一般に用いられている。SMFの詳細は省略するが、個々の演奏情報は、図2に示すように、デルタタイムとイベントという2つの部分で構成される。デルタタイム（ Δt ）は、隣り合ったイベント間の時間間隔を表している。イベントは、音程や音の強さといった種々の演奏情報を含んでいる。この形式は、記憶容量の効率的利用という点では、必ずしも適当なものではない。この理由は大きく分けて

2つある。

【0003】 1つは、イベントデータとして、MIDIメッセージをそのまま使っている点である。例えば、MIDIメッセージの中に、ある特定の音を止める「ノートオフ」メッセージというものがある。これは、ノートオフであることを示すステータスに1バイト、ノートナンバー（音程）を表すのに1バイト、ベロシティ（鍵盤を離す速さ）に1バイトの合計3バイトで構成される。しかしノートオフのベロシティは実際には一定値であることがほとんどで、この場合1つのノートオフメッセージにつき1バイト無駄な情報が保存されていることになる。

【0004】 もう1つは、1つの楽曲の中で同じパターンが繰り返しある場合でも、繰り返し情報を記述しないで、同じデータを重複して記録している点である。通常の楽曲であれば、同じパターンが繰り返される確率は非常に高いので、重複するデータを省くことができれば、記憶容量の有効利用を図ることができる。なお、音符、記号よりなるメロディの楽譜データについて繰り返し部分を効率よく処理する技術が特公昭61-91697号公報に示されているが、これはMIDI信号を前提とするものではなく、通信カラオケに適用することはできない。さらに特開平4-147192号公報には楽譜の繰り返し部分をパターン化する技術が示されているが、十分なデータの圧縮は期待できない。

【0005】

【発明が解決しようとする課題】 SMFには以上のような問題点があるので、特にMIDIデータを大量に保存する音楽データベースにおいて、記憶容量を削減できる保存形式への要求が高かった。また、通信カラオケ等ではデータ量の削減による通信時間及び通信コストの削減が望まれていた。したがって、本発明は時間情報とイベント情報を有する演奏情報データの性質を利用して、データ量を大きく削減することのできる演奏情報圧縮装置を提供することを目的とする。

【0006】

【課題を解決するための手段】 上記目的を達成するために、本発明では演奏情報に含まれる時間情報及びイベント情報の中から同一の情報の出現頻度を検出して符号化するようにしている。すなわち、本発明によれば、楽音を構成する音の発音時間情報と共に音の音程その他を指定するイベントを含む演奏情報に
1 応答し、前記演奏情報のイベント間の相対時間を分析する時間分析手段と、前記演奏情報に
2 応答し、イベントの種類を分析するイベント分析手段と、前記演奏情報に
3 応答し、所定時間間隔でデータを分割してセグメント化し、各々のセグメントから同一内容のデータを検出する楽曲構造分析手段と、前記時間分析手段と前記イベント分析手段と前記楽曲構造分析手段のそれぞれの分析結果に基づいて前記演奏情報を符号化する符号生成手段とを、有する演奏情報圧縮装

置が提供される。

【0007】

【実施例】以下、図面と共に本発明の演奏情報圧縮装置の好ましい実施例について説明する。図1はかかる実施例のブロック図であり、この演奏情報圧縮装置はトラック分離手段10と時間分析手段12とイベント分析手段14と楽曲構造分析手段16と符号生成手段18で構成される。これらの手段は全体を1つ又は複数のCPU

(中央演算処理装置)で構成することができる。入力データはSMFであり、前述したように個々の演奏情報は、デルタタイムとイベントの2つの部分で構成されている。一般に入力データには複数のMIDIチャンネルの演奏情報が含まれている。まずトラック分離手段10において、1トラックに1つのMIDIチャンネルの演奏情報のみ含まれるように、入力データを分離する。入力データがSMFのフォーマット1のようにあらかじめこの条件を満たしている場合は、この処理を省略することができる。

【0008】時間分析手段12はデルタタイムを分析するもので、ここでは入力データに含まれるデルタタイムの種類とその使用頻度を分析する。デルタタイムは、隣り合ったMIDIイベントの間隔をある時間単位で表したものである。現在市販されているMIDIデータ作成ソフトにおいては、4部音符の480分の1を基本単位(1 tick)とするものが多いので、以下では4部音符の480分の1を単位とした場合を例にして説明する。

【0009】図3は、音符の例を示すもので、このような音符の並びをSMFで表した場合、デルタタイム及びデルタタイムの記述に必要なバイト数は表1のようになる。時間分析手段12においては、トラック内のデルタタイムの使用頻度(あるいは、全トラックを通してのデルタタイムの使用頻度)を調べ、表2に示すように、使用頻度の高い順に表2のようなインデックスをつけたデルタタイム頻度表を作り、これを符号生成手段18に出力する。これは非常に単純な例であるが、通常の楽曲においても使用頻度の高いデルタタイムの数は限られたものになる。使用頻度の高いデルタタイムを短い符号に変換すれば、データ量が削減できる。

【0010】

【表1】

デルタタイム	イベント	バイト数
0	ノートオン	1
480	ノートオフ	2
0	ノートオン	1
240	ノートオフ	2
0	ノートオン	1
240	ノートオフ	2
0	ノートオン	1
240	ノートオフ	2
0	ノートオン	1
120	ノートオフ	1
0	ノートオン	1
120	ノートオン	1

【0011】

【表2】

インデックス	デルタタイム	使用回数
0	0	6
1	240	3
2	120	2
3	480	1

【0012】SMFにおいてデルタタイムは可変長符号で表され、値が小さいほど必要なバイト数は少なくて済む。デルタタイムの基本単位が細かいほど音楽的な表現力は高いと言えるが、それに従って必要なバイト数も増える。一方実際に楽曲に使われているデルタタイムを調べると、基本単位の1刻み(1 tick)まで使っていない場合がある。この例では、120 tickが実際に使用されている最小単位である。そこでデルタタイムの最大公約数を求め、符号生成手段18において、個々のデルタタイムをこの最大公約数で除算して符号化する。

【0013】次にイベント分析手段14について説明する。イベント分析の処理としては、全てのイベントについて使用頻度を算出する場合と、特に使用頻度の高い「ノートオン」(ある音を出す)、「ノートオフ」(ある音を止める)イベントに絞って使用頻度を算出する場合の2つがあるが、以下ではノートオンとノートオフに絞った場合を中心に説明する。

【0014】SMFにおいては図4に示すように、ノートオンイベントを3バイト、ノートオフイベントを3バイトで表している。各々のパラメータは、ノートナンバ(音程)とベロシティ(強さ)の2バイトである。

【0015】1曲の中で使われるノートナンバとベロシティの組み合わせは、ある程度限られており、しかも同じ組み合わせが繰り返して使われることが多い。したがって、ノートナンバとベロシティの組み合わせの使用頻度を調べ、多く使われるものを短い符号に変換すれば、

データ量が削減できる。

【0016】イベント分析手段14においては、図5に示す手順でノートオン、ノートオフ各々について、ノートナンバとペロシティの組み合わせの使用頻度をトラック単位あるいは全トラックを通して算出し、使用頻度の高い順にインデックスをつけたノートオン頻度表とノートオフ頻度表をトラック毎にあるいは、全トラックについて1つ作成し、符号生成手段18に出力する。表3に図3の音符を例とした場合のノートオン頻度表を示す。*

インデックス	ノートナンバ	ペロシティ	使用回数
0	60	96	3
1	64	96	2
2	64	80	1

* ノートオフ頻度表も同様である。以上の実施例では、ノートオンとノートオフのみを対象として、各々の使用頻度を算出したが、全ての種類のイベントを対象にした使用頻度を算出する場合は、ノートオン頻度表、ノートオフ頻度表の代わりに表4のようなイベント頻度表を作ればよい。

【0017】

【表3】

【0018】

※ ※ 【表4】

インデックス	使用頻度	データ長	データ
0			
1			
.....			
N			

【0019】ところで、音程を連続的に変えるイベントであるピッチベンドなどは、SMFにおいてパラメータが少しずつ変わりながら連続して使われていることが多い。このような部分を以下では、連続イベント部分と呼ぶが、連続イベント部分を検出し、その最初と最後の位置を記録して、符号生成手段18に出力する。この情報の利用については後述する。

【0020】次に楽曲構造分析手段16について説明する。まず演奏情報を一定の時間間隔（例えば、1小節相当）で分割する。分割された要素をここではセグメントと呼ぶ。前述したように、通常の楽曲では、同じ内容のセグメントが繰り返して使われている場合が多いので、このようなセグメントの検出を行う。

★ 【0021】セグメントには、楽曲の先頭から順に0, 1, 2...Nといった番号を付け、セグメントの先頭位置及び長さの情報と共にセグメント表として記述する。先頭位置は入力データの先頭からのオフセットバイトで表し、長さはセグメントのバイト数で表す。i番目のセグメントに対し、0～i-1番目のセグメントとの比較を行い、同じ内容のセグメントがあれば、最も若い番号をセグメント表に記録する。このような処理をi=1～Nとして行った後、セグメント表を符号生成手段18に出力する。セグメント表の一例を表5に示す。

【0022】

【表5】

セグメント番号	先頭位置	長さ	同じ内容のセグメント
0	0	100	なし
1	100	120	なし
2	220	100	1
3	320	120	2
4	440	100	なし
5	540	120	2
6	660	120	5

【0023】次に符号生成手段18について説明する。本装置で生成する符号は図6の(a)に示すように、ヘッダ、デルタタイム、ノートオンマップ、ノートオフマップ、データブロックの各部分で構成されている。デー

タブロックの要素は、図7に示すように、mビットのデルタタイムフラグとnビットのイベントフラグと可変長のデータ部で構成される。

【0024】符号生成手段18では、トラック分離手段

10の出力を順次走査し、デルタタイムとイベントを順次読み込む。その際、楽曲構造分析手段16で作成されたセグメント表を参照して、現在読み込んでいるデータがどのセグメントに属しているか、またそのセグメントと同じセグメントの有無についてチェックする。セグメント表に同じ内容のセグメントがある場合は、イベントフラグにセグメント参照を意味するコードをセットし、データ部に参照するセグメント番号を出力する。そして次のセグメントが始まるまでトラック分離手段10から出力を読み飛ばす。セグメント表に同じ内容のセグメントが無い場合は、イベントフラグにセグメント定義を意味するコードをセットする。そのセグメントについて以下の処理を行った後、セグメント定義終了を意味するコードをイベントフラグにセットする。以上セグメント記述をまとめると図10のようになる。

【0025】セグメントを定義する場合の処理は次のようになる。デルタタイムの値が、前記デルタタイム頻度表の先頭から $2^m - 1$ 個の範囲にあれば、デルタタイムフラグに、そのインデックスを記述する。この場合、データ部にはデルタタイムのデータは出力しない。またこの範囲になれば、 m ビットを全て1にして、データ部

【0026】イベントフラグは n ビットあるので、 2^n 種類のイベントが記述できるが、マップにないことを示すのに1個、セグメントの記述用に3個必要なので、ノートオン用に p 個、ノートオフ用に q 個、連続イベント記述用に r 個のコードを割り当てる(ただし、 $p + q + r < 2^n - 4$)。イベントがノートオンである場合は、ノートオン表を参照し、インデックスが0から $p - 1$ の範囲で、ノートナンバとベロシティが一致すれば、そのインデックスをイベントフラグにセットする。この場合は、データ部に出力しない。もしノートオン表のその範囲になれば、 n ビットを全て1にして、データ部にSMFと同様のイベント情報を記述する。ノートオフについても同様である。

【0027】イベント分析手段14で検出した連続イベント部分を読み込んだ場合、以下の処理を行う。イベントフラグには、連続イベント用に割り当てられた r 個のコードの中から、イベントの種類に対応したものを選び、それを設定する。データ部は、図8の(a)で示したフォーマットとする。

【0028】図9に示した連続イベントを例に取って、具体的に説明する。連続イベント部分は、この例のようにほぼ同じ時間間隔でパラメータ値が少しずつ変化していることが多い。この例では、ピッチベンドイベントが6個あり、イベントの時間間隔は全て50、パラメータ値は各々10、11、12、15、18、21となっているものとする。SMFで記述する場合は、各々のデルタタイムに1バイト、イベントに3バイト必要として、

$6 \times (1 + 3) = 24$ バイト必要である。この例を図8の(a)に従って、実際に記述したものが図8の(b)である。8バイトで記述ができ、SMFに比べて大幅に記憶容量が削減できる。

【0029】ヘッダには、図6の(b)に示すように、トラックの長さ、デルタタイムの最大公約数が記述される。デルタタイムマップは図6の(c)に示すように、マップに含まれるデルタタイムの数と、時間分析手段12で作成したデルタタイム頻度表の中のインデックスが0から $(2^m - 2)$ を超えない範囲のデルタタイムを順番に並べたものである。

【0030】ノートオンマップは図6の(d)に示すように、マップに含まれるデータ数と、イベント分析手段14で作成したノートオン頻度表のインデックスが0から $p - 1$ までのノートナンバとベロシティを順番に並べたものである。ノートオフマップも同様であるが、ノートオンとは違いノートオフのベロシティは全て同じ値になっていることが多いので、ベロシティ情報を記述する1バイトを追加する。ノートオフベロシティが全て同じ値であれば図6の(e)に示すように、その値(0~127)をこの場所に記述し、続いてノートオフ頻度表のノートナンバのみ q 個並べる。もし一定値でなければ、図6の(f)に示すように先頭の1バイトに128以上の適当な値を書き込み、続いてノートナンバとベロシティの組を q 個並べる。なお、この例では、デルタタイムマップ、ノートオンマップ、ノートオフマップをトラック毎に作成したが、これらを全トラックに対して1つつ用意してもよい。

【0031】以上本発明の演奏情報圧縮装置について説明したが、図3の音符の並びが具体的にどのような符号に変換され、どの程度圧縮されるかを示す。ノートオフベロシティは全て64で一定とする。SMFで記述する場合、トラックの長さを表すのに4バイト、表1に示したようにデルタタイムに16バイト、ノートオン6回、ノートオフ6回各々3バイトずつとして36バイト、合計56バイト必要である。

【0032】一方 $m = 3$ 、 $n = 5$ 、 $p = 3$ 、 $q = 2$ とした場合の本装置の出力はヘッダ5バイト、デルタタイムマップ5バイト、ノートオンマップ7バイト、ノートオフマップ4バイト、データブロック12バイトで合計33バイトとなり、データ圧縮される。この例はデータ数が少なく、セグメントの重複が無い例であるが、一般の楽曲ではセグメントの重複があるので更に効率的に圧縮できる。

【0033】ノートオン頻度表、ノートオフ頻度表の代わりにイベント頻度表を使う場合は、出力は、図6の(g)に示すような構成となる。イベントマップは図6の(h)に示すように、イベントマップに含まれるデータの個数に続き、イベント頻度表のデータ長とデータの組を $2^n - 1$ 個超えない範囲で並べる。また、データ

10

20

30

40

50

ロックのイベント記述は、 $2^n - 1$ 個のコードを使って上記と同様に行なえば良い。

【0034】なお、以上詳述したフォーマット並びに処理手順は一例であり、その主旨を逸脱しない範囲において種々の変更を加えることができる。

【0035】

【発明の効果】以上説明したように本発明の演奏情報圧縮装置は上記構成なので、原データの持つ情報を全く失うことなく、大幅なデータ削減が可能であり、特にMIDIデータを大量に保存する音楽データベースにおいて、記憶容量が少なくすみコストが削減できる。またMIDIデータを通信回線で送る場合には、送信時間が短縮でき、コストも削減することができる。

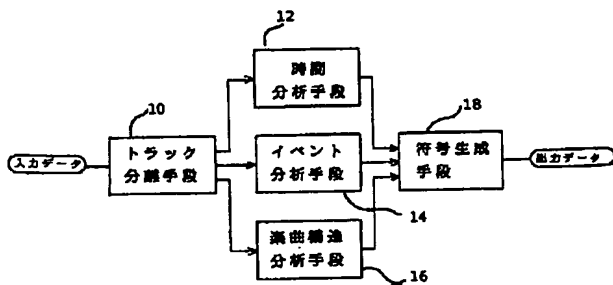
【図面の簡単な説明】

【図1】本発明の演奏情報圧縮装置の好ましい実施例のブロック図である。

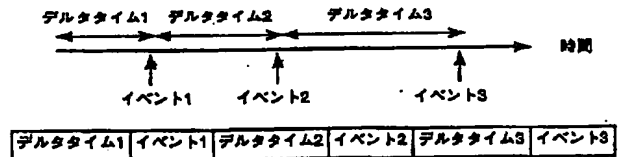
【図2】SFMデータの説明図である。

【図3】演奏データの一例を示す図である。

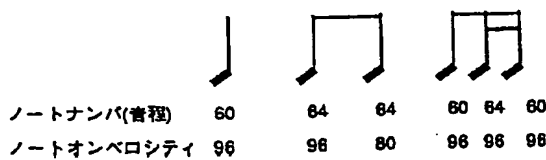
【図1】



【図2】



【図3】



【図4】

1バイト目	2バイト目	3バイト目
ノートオン ステータス	ノートナン バ	ペロシティ
ノートオフ ステータス	ノートナン バ	ペロシティ

【図7】

デルタフラグ (mビット)	イベントフラグ (nビット)	データ部(可変長)
------------------	-------------------	-----------

【図4】SMFのノートオン、ノートオフの構成図である。

【図5】イベント分析手段の動作を示すフローチャートである。

【図6】本装置の出力する符号の構成を示す図である。

【図7】データブロックの要素の説明図である。

【図8】連続イベントを記述するフォーマットを示す図である。

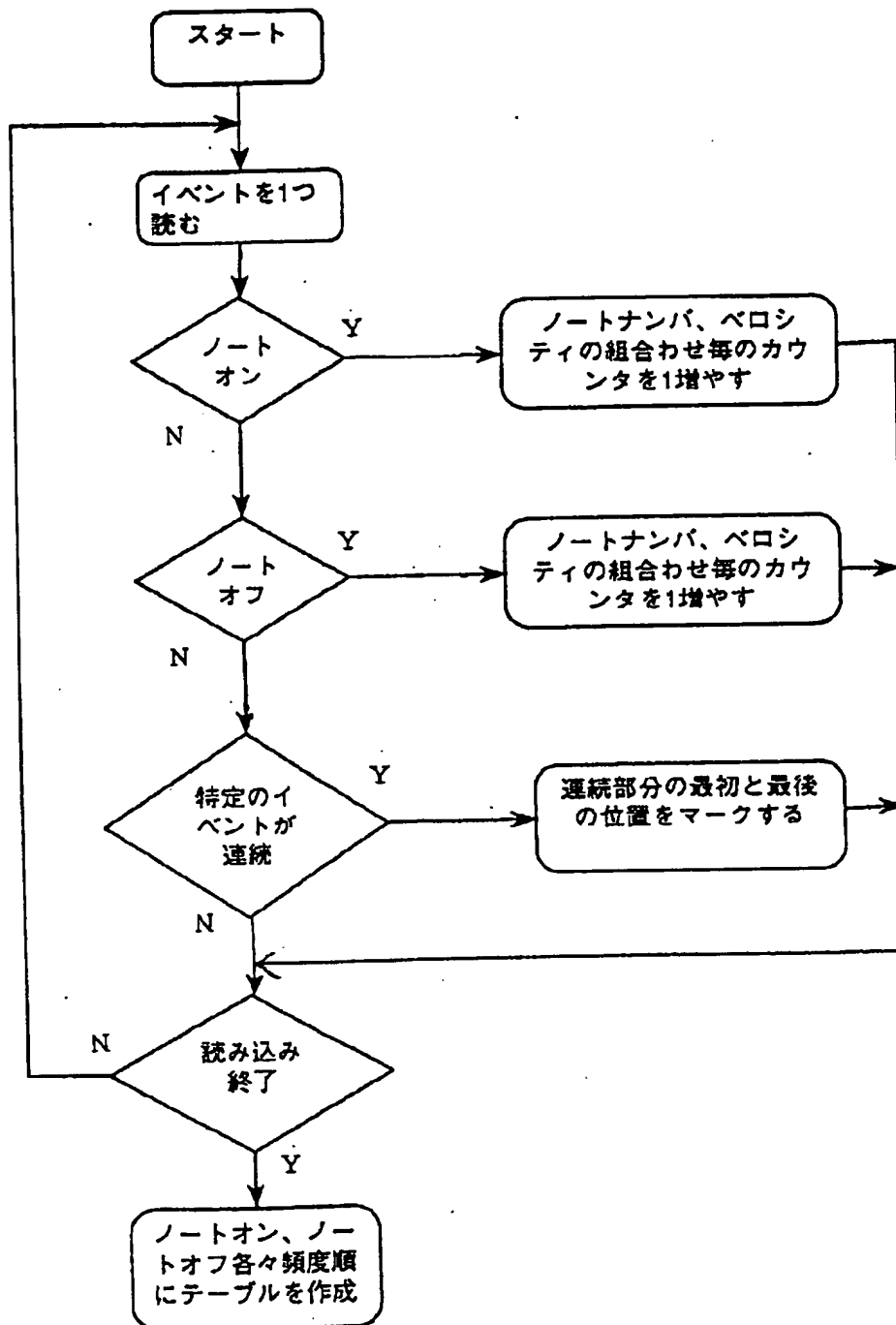
【図9】連続イベントの一例を示す図である。

【図10】セグメントを記述するフォーマットを示す図である。

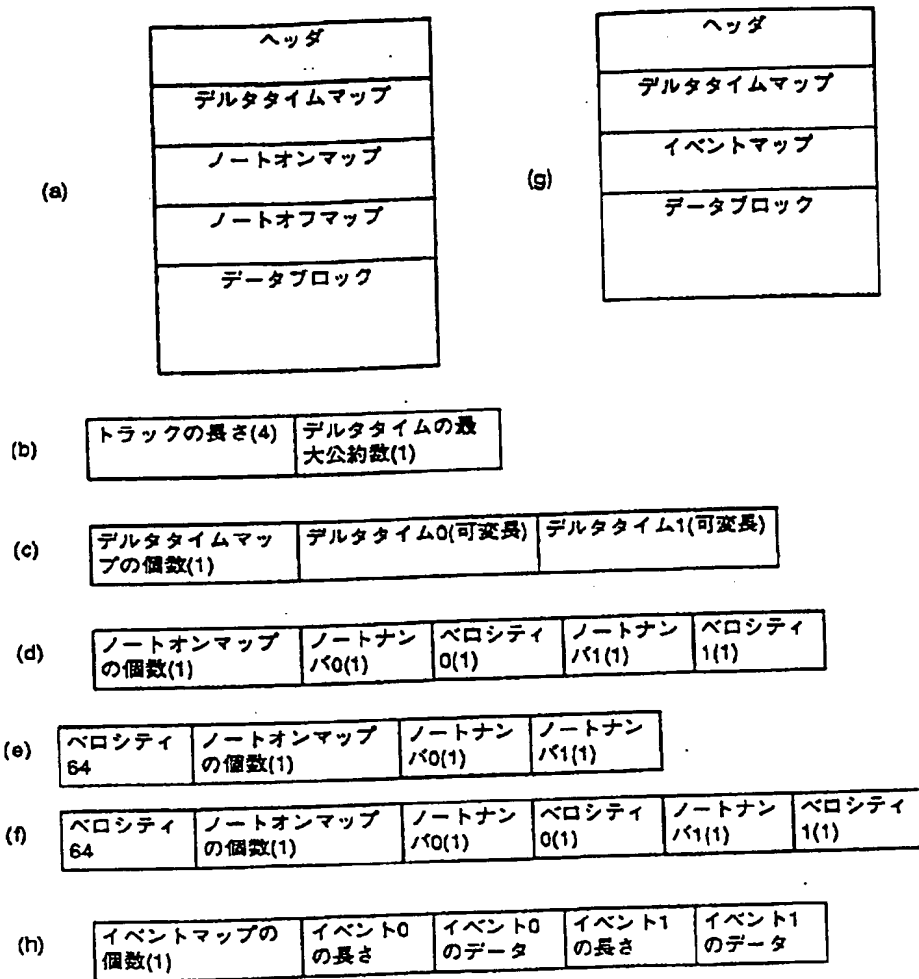
【符号の説明】

- 10 トラック分離手段
- 12 時間分析手段
- 14 イベント分析手段
- 16 楽曲構造分析手段
- 18 符号生成手段

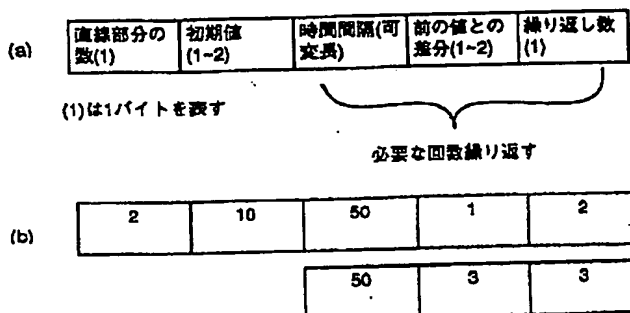
【図5】



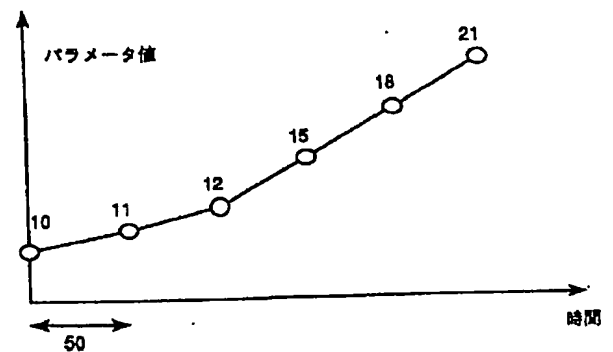
【図6】



【図8】



【図9】



【図10】

同じ内容のセグメントを参照する場合

デルタタイムフラグ	イベントフラグ	データ部	
		セグメント参照コード	セグメント番号

セグメントを定義する場合

デルタタイムフラグ	イベントフラグ			デルタタイムフラグ	イベントフラグ
		セグメント定義コード	そのセグメントの符号		定義終了コード